NASA Technical Memorandum 102455

Concurrent Processing Adaptation of Aeroelastic Analysis of Propfans

David C. Janetzke
Lewis Research Center
Cleveland, Ohio

and

Durbha V. Murthy
University of Toledo
Toledo, Ohio

January 1990

NASA

CONCURRENT PROCESSING ADAPTATION OF AEROELASTIC ANALYSIS OF PROPFANS

David C. Janetzke
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

and

Durbha V. Murthy*
University of Toledo
Toledo, Ohio 43606

Abstract

This paper reports on a study involving the adaptation of an advanced aeroelastic analysis program to run concurrently on a shared memory multiple processor computer. The program uses a three-dimensional compressible unsteady aerodynamic model and blade normal modes to calculate aeroelastic stability and response of propfan blades. The identification of the computational parallelism within the sequential code and the scheduling of the concurrent subtasks to minimize processor idle time are discussed. Processor idle time in the calculation of the unsteady aerodynamic coefficients was reduced by the simple strategy of appropriately ordering the computations. Speedup and efficiency results are presented for the calculation of the matched flutter point of an experimental propfan model. The results show that efficiencies above 70 percent can be obtained using the present implementation with 7 processors. The parallel computational strategy described here is also applicable to other aeroelastic analysis procedures based on panel methods.

---

*NASA Resident Research Associate at Lewis Research Center (work funded under NASA Grant NAG3-742).

Introduction

It is now clear that the analysis and design of practical engineering systems in the future will make extensive use of concurrent processing on computers with multiple processors. Parallel processor computers consisting of multiple processors linked to one another and running subtasks concurrently are becoming widely available for large scale scientific and engineering computations (Noor and Atluri, 1987). These computers are being increasingly used to provide the reduction in effective calculation time (or elapsed time) needed to carry out many real time applications. A new area of research in computational mechanics is based upon the unique architectural features of parallel processing computers.

Aeroelastic analysis of practical aerospace structures has always been a computationally expensive process. The computational burden is particularly high for aeroelastic analysis of propfans because of their very complicated geometry (Kaza, et al 1989; Kaza, et al 1988; Murthy and Kaza, 1989). Automated design and optimization of aerospace structures requires the aeroelastic analysis to be performed many times inside a design iteration loop and it is expected that conventional sequential processing computers would be inadequate for obtaining useful results in a reasonable time. The computational power of the new parallel processing computer systems offers the aeroelastician an opportunity to reduce computational times for analysis so that the aeroelastic analysis can be more fruitfully used by the designers. This paper presents a concurrent processing adaptation of an advanced aeroelastic analysis code for propfans. The emphasis is on the concurrentization of the computationally intensive portion of the aeroelastic analysis in order to reduce the effective calculation time.

Because of the promise of considerable fuel savings over current turbofan engines at similar flight speeds, propfans are expected to be the preferred propulsion

system of the next generation of transport aircraft. Avoidance of aeroelastic instabilities is a major part of the design process of propfans. Prediction of the forced response of propfans is also an important concern. As part of an effort to study and predict these instabilities and the forced response of the propfan blades, an aeroelastic analysis program called ASTROP3 was developed at the NASA Lewis Research Center. This program was developed for and was operational in a sequential processing environment before it was specifically adapted and implemented on a parallel processing computer.

The ASTROP3 aeroelastic analysis code was adapted to concurrent processing on a multiple instruction multiple data (MIMD) parallel computer with shared memory, i. e., having asynchronous multiple processors. By asynchronous operation, it is meant that different sets of instructions are executed concurrently and independently start or finish. In the shared memory architecture, all the processors have access to a common global memory. An alternative architecture of a large number of processors having attached local memories, with data being passed from processor to processor as needed, is also of great importance. However, the authors anticipate that most of the near-future multiple processor commercial computers will have shared memory architecture. For example, the commercial success of the CRAY-XMP, CRAY-2 and ALLIANT computers indicates this trend.

Approaches to Sequential Code Conversion

Given the description of a problem and a computer program which solves the problem sequentially, optimal gains in performance through parallelism could be attained perhaps only by a complete reformulation of the solution method from scratch and a redesign of the computational algorithm to take advantage of the parallel architecture. The new computational algorithm would typically maximize the computations that can be done in parallel and minimize those that must be done

sequentially. It would explicitly consider the relationship between the size of the problem and the number of available processors and optimize the balance of workload among the processors and minimize synchronization delays. This requires a thorough understanding of the nature of the problem and a considerable investment in analyzing the data flow in the solution procedure itself rather than its implementation in the sequential code. Such an operation is usually prohibitively expensive and is not likely to be undertaken extensively because of the substantial investment in many existing sequential computer programs. It may also be unnecessary because significant gains in performance can often be obtained by only a moderate modification of the sequential code.

Parallelism can be implemented either at a low-level (fine grain parallelism) or at a high-level (coarse grain parallelism). A low-level approach introduces concurrency at the level of elementary operations. It is easier to implement when powerful compilers that automatically parallelize code are available as on the Alliant FX/80. The high-level approach introduces concurrency in a top down manner and generally requires manually changing the computational strategy of the sequential program and the disabling of the automatic parallelization done by the compiler. The high level approach is natural when the physics of the problem possesses natural parallelism in the computations. It has the added advantage that the performance improvements achieved by parallelism are, in a sense, machine-independent. For example, when a high-level approach is used, the computational strategy developed for a multiple processor environment can be transported to a multiple computer environment with interconnections as in multiple workstations on a network. One disadvantage of this approach is the increased risk of errors due to the disabling of the conservative parallelization by the compiler and the manual analysis of the data dependencies among the parallel paths.

For ease of implementation and to serve as a learning exercise, we selected a high-level approach requiring only a moderate modification of the sequential code. The key to the high-level approach is the identification of computationally intensive parallel paths in the sequential program and the implementation of concurrent computation with a minimum of data dependency among the individual subtasks. The goal is to reduce the calculation time in proportion to the number of processors, i.e., to achieve linear speedup, as closely as possible. The key to achieving this reduction is the selection of the parallel paths in the numerical algorithm. Because there is some cost to concurrentization, a parallel processing implementation gives the most benefit when the computational time of each of the parallel paths is large and the data dependencies among them are small. Thus the formulation and the computational strategy of the sequential code must permit the decomposition of the problem solution into concurrently executable processes (or subtasks), each of which is computationally intensive and is fairly independent of the data computed by the other processes. In this paper, it is shown that the computation of aerodynamic influence coefficients, commonly used in the computation of unsteady generalized forces in aeroelastic analyses, fulfills these requirements and thus can be parallelized easily and in a natural manner.

Propfan Aeroelastic Analysis

The aeroelastic analysis code used in this study is a FORTRAN program called ASTROP3 and was developed at NASA Lewis Research Center for single rotation propfans(Kaza et al 1989). This program performs flutter and forced response analysis of propfans. The implemented version of ASTROP3 incorporates the enhancements developed by Murthy and Kaza (1989) but is only valid for subsonic flow. In the following, the analytical procedure used in ASTROP3 is briefly reviewed to aid in the understanding of the adaptation for concurrent processing.

The propfan is assumed to have identical groups of blades symmetrically distributed about a rigid disk rotating at a fixed speed Ω in an axial flow of Mach number M. The linearized aeroelastic equations of motion of the rotor are then uncoupled for different inter-group phase angle modes. Assuming simple harmonic vibratory motion with a frequency ω, and expressing the vibratory motion in each inter-group phase angle mode as a superposition of assumed mode shapes, the equations of motion for a given inter-group phase angle σ, can be written in the form (Kaza et al 1989)

$$[-\omega^2\mathbf{M}_g + \mathbf{K}_g]\mathbf{q} = [\mathbf{A}(M, \omega)]\mathbf{q} + \mathbf{f} \tag{1}$$

where $\mathbf{M}_g$, $\mathbf{K}_g$ and $\mathbf{A}(M, \omega)$ are the generalized mass, stiffness and aerodynamic matrices respectively, $\mathbf{f}$ the generalized aeroelastic forcing vector and $\mathbf{q}$ the generalized coordinate vector. The matrix $\mathbf{A}$ represents the motion-dependent aerodynamic forces and the vector $\mathbf{f}$ the motion-independent aerodynamic forces. The matrices in eq. (1) are of order nm where n is the number of blades in each group and m is the number of assumed structural mode shapes per blade. The aerodynamic matrix $\mathbf{A}(M, \omega)$ is usually valid only for simple harmonic motion of the airfoil.

The matrices, $\mathbf{M}_g$ and $\mathbf{K}_g$ are obtained using MSC/NASTRAN, a general purpose finite element program. $\mathbf{A}(M, \omega)$ and $\mathbf{f}$ are obtained using the unsteady aerodynamic model developed by Williams and Hwang (1986) for subsonic axial flow. The computational procedure in the unsteady aerodynamic model is briefly reviewed here as it is necessary to understand the results presented later. Complete description of the procedure is found in Williams and Hwang(1986).

The unsteady aerodynamic forces are calculated by integrating the unsteady pressure disturbance over the blade surface. The unsteady pressure disturbances and the normal velocity over a thin blade are related by an integral equation.

Assuming simple harmonic motion with a frequency ω in a three-dimensional potential flow, this integral equation can be written, after appropriate linearization, as

$$W(P) = -B\int_A \Delta\bar{p}(P_0)\frac{\partial}{\partial\bar{\theta}}[K(P,P_0)]dA_0 \tag{2}$$

where P and P_0 represent points on the blade surface, A represents the blade surface, ω is the frequency of blade vibration, B is a constant dependent on flow conditions and K is a kernel function. W and $\Delta\bar{p}$ are proportional to the normal velocity and unsteady pressure disturbance respectively.

Eq. (2) was discretized by splitting the blade into n_p quadrilateral panels within each of which $\Delta\bar{p}$ is assumed constant. (See Figure 1). The discretization results in the algebraic system of equations given by

$$\mathbf{W} = \mathbf{C}\Delta\bar{\mathbf{p}} \qquad \text{Or, } \Delta\bar{\mathbf{p}} = \mathbf{C}^{-1}\mathbf{W} \tag{3}$$

where $\mathbf{W}$ is a vector of the values of W at chosen control points on each of the panels, $\Delta\bar{\mathbf{p}}$ is a vector of the values of $\Delta\bar{p}$ on each of the panels, and $\mathbf{C}$ is a matrix of aerodynamic influence coefficients given by

$$c_{ij} = -\int_{A_j}\frac{\partial}{\partial\bar{\theta}}[K(P_i,P_0)]dA_0 \tag{4}$$

where the subscripts i and j refer to the control panel and pressure panel respectively. In terms of radial coordinate r and chordwise azimuthal coordinate $\bar{\theta}$, (Figure 1), eq. (4) can rewritten as

$$c_{ij} = -\int_{r_{1j}}^{r_{2j}} \int_{\bar{\theta}_{jLE}}^{\bar{\theta}_{jTE}} \frac{\partial}{\partial \bar{\theta}_0} \left[K(\bar{\theta}_i - \bar{\theta}_0, r_i, r_0) \right] r_0 d\bar{\theta}_0 dr_0 \tag{5}$$

The subscripts *jLE* and *jTE* refer to the leading and trailing edges of the *j*-th panel. The chordwise integration in eq. (5) can be performed analytically, so that

$$c_{ij} = D_{ijLE} - D_{ijTE} \tag{6}$$

where

$$D_{ijLE/TE} = \int_{r_{1j}}^{r_{2j}} K(\bar{\theta}_i - \bar{\theta}_{0LE/TE}, r_i, r_0) dr_0 \tag{7}$$

Thus, the evaluation of the influence coefficient c_{ij} requires numerical integration of the kernel function K in the radial direction only. The kernel function must itself be evaluated by numerical integration. We refer to r_i as the control panel row radius and r_0 as the pressure panel row radius. Because of the analytical integration in the chordwise direction, the computational effort is nearly independent of the number of chordwise panels and is approximately proportional to the square of the number of radial panel rows.

Once the influence coefficients are evaluated, the generalized motion-dependent force matrix is determined by numerical integration over the blade surface

$$A_{nm} = \sum_{j=1}^{n} \Delta p_{jm} \delta_{jn} \, dA_j \tag{8}$$

where Δp_{jm} is the pressure differential across the j -th panel, with area dA_j, due to motion in the m-th assumed mode shape, and δ_{jn} is the normal displacement of the blade surface at the control point of the j -th panel in the n-th mode shape. The motion-independent aerodynamic force vector is similarly calculated by

$$f_n = \sum_{j=1}^{n} \Delta p_{jF} \delta_{jn} \, dA_j \tag{9}$$

where the subscript F represents the assumed forcing distribution.

The flutter problem for the inter-group phase angle of interest, is solved by setting the motion-independent aerodynamic force vector **f** to zero, and finding the combination of M and ω for which eq. (1) has a nontrivial solution. That is, we solve the characteristic equation

$$\det[-\omega^2 \mathbf{M}_g + \mathbf{K}_g - \mathbf{A}(M, \omega)] = 0 \tag{10}$$

for M and ω. The numerical algorithm for solving eq. (10), given the initial guesses for the critical Mach number and frequency, is described by Murthy and Kaza (1989) and involves evaluating the aerodynamic matrix a number of times to find the critical Mach number and frequency. The critical Mach number for the propfan is then the lowest Mach number at which one of the inter-group phase angle modes flutters.

The forced response problem is solved by evaluating the aerodynamic matrix at the axial flow Mach number and excitation frequency and the excitation inter-group phase angle (Kaza et al 1988) and solving eq. (1) for the generalized coordinates, **q**.

The results presented in this paper were obtained using the SR3C-X2 propfan rotor with eight identical blades for flutter analysis. The rotational speed was fixed at 5280 rpm. This rotor was earlier analyzed using ASTROP3 and results were

reported by Kaza et al (1989) and Murthy and Kaza (1989). The motion of the blade is dominated by the first two normal modes of the blade, though up to six normal modes were used in some of the analysis cases. The blade surface was discretized such that there were 8 panels in the chordwise direction in each radial row. The number of panel rows in the radial direction was either 9 or 17. Thus the total number of panels was 72 or 136 respectively. The results presented are for the critical inter-group phase angle, which at this rotational speed was 225°. The critical Mach number and frequency calculated were in the range 0.62-0.65 and 287-295 Hz respectively, depending on the number of panels and number of modes used in the analysis. Unless otherwise mentioned, initial guesses of $M = 0.5$ and $\omega = 310$ Hz are used.

Identification of Parallelizable Code

The determination of the aerodynamic influence coefficient matrix **C** requires the evaluation of the kernel function repeatedly for different control panel rows and pressure panel rows on the blade surface and its radial integration as indicated in eq. (7). In both flutter and forced response problems, the computation of the kernel function is very expensive as it involves wake integration which requires the numerical evaluation of an integral with an infinite limit. Thus, the kernel function computation and hence the evaluation of the aerodynamic influence coefficients is the dominant contributor to the calculation time required to compute the generalized unsteady aerodynamic forces. For example, in the case of the SR3C-X2 propfan blade using 9 radial panel rows and 2 normal modes, the computation of the aerodynamic influence coefficients consumes 97.1 percent of the time required for the calculation of the critical Mach number and frequency starting with a moderate initial guess of $M = 0.5$ and $\omega = 310$ Hz . The computational expense of evaluating the kernel function by direct numerical integration had prompted several researchers to develop

and study efficient and approximate methods of integration. Desmarias (1982) assessed the performance of many such methods.

Fortunately, the computation of the influence coefficients possesses a high degree of parallelism so that an aeroelastic analysis program using a parallel processing computer would greatly benefit from concurrentization of their computation. Simply put, the kernel function $K(\overline{\Delta\theta}_i, r_i, r_0)$ in ASTROP3 is evaluated by interpolation after constructing a table of values for K at various values of $\overline{\Delta\theta}_i$ and fixed values of r_i and r_0. The number of values of r_i and r_0 is given by the number of panel rows used along the radial direction in the discretization of the planform. Once the blade geometry, motion and the flow conditions are given, the table of values of $K(\overline{\Delta\theta}_i, r_i, r_0)$ at various values of $\overline{\Delta\theta}_i$ for fixed values of r_i and r_0 can be computed independently of the value of the kernel function at other values of r_i and r_0 . Thus, the kernel function $K(\overline{\Delta\theta}_i, r_i, r_0)$ can be evaluated completely in parallel for different values of r_i and r_0. The radial integration of eq. (7) can also be performed in parallel for different values of r_i and r_0.

There are of course other parts of the analysis that can also be done concurrently, e. g., the computation of the normal velocities for motion in different mode shapes. However, in this implementation, only the influence coefficient computation was concurrentized because the bulk of the computational effort is expended there. Concurrentizing the other parts of the analysis would have required more effort with little potential gain in speedup and efficiency.

Parallel Processing System Description

The parallel processing computer used in this study was an Alliant FX/80. The FX/80 is classified as a multiple instruction, multiple data (MIMD) computer with shared memory. It has eight general purpose 64-bit vector processors and twelve

general purpose 32-bit scalar processors. The vector processors can work together to provide concurrent processing on a group of subtasks within a single task. The scalar processors independently execute interactive user jobs such as editors and operating system tasks. All processors share a 128 Mb global memory through a high-speed cache system.

A simplified schematic of the system architecture as it pertains to a single parallel processing task is given in Figure 2. During the course of this study, seven of the vector processors were configured to run parallel processing tasks. The remaining vector processor is used on non-parallel tasks such as compilations and operating system tasks. Any number of the seven vector processors can optionally be used on a parallel processing task. The operating system for the FX/80 is based on Berkeley UNIX with extensions for parallel processing.

The FORTRAN compiler for the FX/80 can automatically optimize standard FORTRAN code for scalar, vector and concurrent processing. The compiler does a data dependency analysis to ensure valid results from the optimization. These optimizations are selected with options in the compile statement. Any, all or none of the optimizations can be selected. Additionally, compiler directives may be inserted in the source code to prohibit or enable specific optimizations in portions of the program.

Computer Implementation

For efficient execution of any analysis procedure into a parallel processing environment, the following requirements must be ensured: (1) the overhead involved in implementing the algorithm must remain as low as possible and (2) the idle time of the available processors must be as low as possible.

Reducing Overhead

Overhead is that time used in extra coding to setup vector processing or concurrent processing. For a given program, such as the ASTROP3 code, a combination of actions is needed to minimize the overhead. These actions include: (1) selective use of compiler options; (2) judicious use of compiler directives; and (3) alterations to the source code. All of these were used to attain efficient execution of ASTROP3 on the Alliant FX/80.

Various combinations of compiler options were applied to the unmodified version of the ASTROP3 program to determine the effect on execution time. The non-selective application of all the compiler optimizations (scalar, vector, and concurrent) to all the routines resulted in execution times on one or more processors significantly (over 30 percent) greater than that for a scalar-only optimized run on one processor. Selective use of vector optimization, guided by familiarity of the code and profiling results, reduced much of the unnecessary overhead and lowered the one-processor execution time significantly (over 20 percent). No reduction in execution time for multiple processors was obtained with concurrent optimization on the unmodified code.

Significantly improved execution times on multiple processors were obtained only after modification of the ASTROP3 source code. The computation of influence coefficients, identified as parallelizable in a previous section, was rendered concurrent by a few small changes and insertion of appropriate compiler directives in the code. This reduced the execution time for 7 processors to under 24 percent of the scalar/vector optimization on one processor.

Reducing Idle Time

To further improve processor utilization, the concurrent subtasks must be scheduled such that the available processors are not left idle for a significantly long time. When asynchronous execution is allowed, as on the FX/80, the processors are left idle only when no additional subtask is available to be initiated, that is, as the concurrent computational phase draws to a close. (The idle time between the completion of a subtask and the initiation of a new subtask was found to be negligible.) If the time required for the execution of each of the subtasks is known a priori, as is usually the case in low-level parallelism, a scheduling algorithm could be devised and processors could be statically allocated so that each subtask is appropriately assigned to a specific processor to minimize the processor idle time. This is known as static load balancing. The assignment of subtasks to processsors at execution time is, on the other hand, known as dynamic load balancing.

In the computation of the aerodynamic influence coefficients, the time required to calculate the coefficients for each combination of pressure panel radius and control panel radius is not known a priori. Hence, static load balancing cannot be used effectively. It was discovered that the computation of the coefficients when control panel row and the pressure panel row coincide is much more time consuming than when they do not coincide. This is illustrated in Figure 3 where the computation time is shown as a function of IRC and IRP, where IRC is the control panel row number and IRP is the pressure panel row number. The computation times for the aerodynamic coefficients when IRC $\neq$ IRP are approximately equal and significantly less than when IRC = IRP.

This knowledge about the relative magnitude of calculation time required for coincident and non-coincident control and pressure panel rows can be used to devise a load balancing scheme that minimizes processor idling. However, on the FX/80, the

user has no control over which processor executes which subtask. Subtasks are selected from a pool of subtasks and assigned sequentially to processors as they become available. However, by specifying the order of execution of the subtasks, some control can be exercised over subtask allocation to processors, thus influencing the load balancing among the processors. Figure 4 illustrates three possible orders for the evaluation of the kernel function prior to the calculation of the unsteady aerodynamic influence coefficients. The numbers inside circles represent the sequence number of evaluation of kernel function for the given values of IRC and IRP in the array. Thus, in evaluation order (a), the kernel function is evaluated by proceeding along the columns. In evaluation order (b), the kernel function is evaluated proceeding along the diagonals from bottom left to top right of the array. Evaluation order (c) is same as (a) except that the last column is reversed. The evaluation order (a) is the original evaluation order used in the sequential code.

Significant idling of the processors is possible if a combination of subtasks, having large differences in computational times among them, is executing near the end of the parallel computational phase. From Figure 3 and the previous discussion, it is clear that this corresponds to a combination of subtasks having coincident and non-coincident control and pressure panel rows executing near the end of the aerodynamic coefficient computation. Then, the subtasks having non-coincident control and pressure panel rows would finish before that having coincident control and pressure panel rows, thus making significant idling of the processors inevitable. This is the case for evaluation order (a) in which the last executing subtask, number 81 in Figure 4(a), would necessarily have IRC = IRP. For evaluation orders (b) and (c), in the subtasks near the end of the parallel computational phase, there would be no subtask for which IRC = IRP, as long as the number of processors is less than the number of radial panel rows. Hence, in the absence of any variation in the shared information access time, the evaluation orders (b) and (c) of Figure 4 would result in significant reduction of idle time compared to the evaluaton order (a).

Table 1 lists the average idle time per processor for the different evaluation orders as a fraction of the total computational time for the aerodynamic coefficients using 7 processors. It is clear that the processor idle time can be significantly reduced by modifying the original evaluation order. However, the effect of modifying the evaluation order is less impressive when the number of panel rows is large. The difference in the idle times associated with the evaluation orders (b) and (c) is considered insignificant and is attributed to small differences in the times required for the calculation of the kernel function for different combinations of IRC and IRP and to differences in shared memory access time due to differences in the order of computation. This difference has very small effect on the speedups associated with the complete flutter analysis.

Speedup and Efficiency Results

The aeroelastic analysis program was implemented on the FX/80 using the evaluation order (c) discussed above. Here, we discuss speedup and efficiency results obtained with this implementation.

The performance of a parallel implementation is usually measured by speedup and efficiency. Speedup is the ratio of time used by the program when executed on a single processor to that used on multiple processors. Efficiency is defined as speedup divided by the number of processors. Speedup is a measure of the reduction in the effective calculation time achieved by the parallel algorithm whereas efficiency is a measure of the processor utilization.

Under ideal circumstances, a perfectly parallelized program will run p times faster on p available processors, than on a single processor. Thus, the ideal speedup is given by

$$S_p = p \tag{11}$$

and the corresponding efficiency would be 100 percent. In practice, however, this is never achieved because 1) there is some overhead associated with concurrent processing and 2) there is always some portion of the program which cannot run concurrently. If α is the fraction of parallel code and p the number of processors, then, under appropriate assumptions (Ortega, 1988), the theoretical speedup is given by

$$S_p = \frac{1}{(1-\alpha) + \alpha/p} \tag{12}$$

For this study, α is the same as the fraction of the total analysis time consumed by the computation of unsteady aerodynamic coefficients. The actual speedup would generally be less than that indicated by eq. (12) because of processor idling and concurrent processing overhead. Thus, the theoretical speedup given by eq. (12) also provides an upper bound on the speedup that can be achieved even when an unlimited number of processors are available. Thus,

$$S_p \leq \frac{1}{1-\alpha} \tag{13}$$

This is known as Amdahl's Law. For example, in the case of the flutter analysis of the SR3C-X2 propfan rotor using 9 panel rows, α was 0.971 as previously mentioned. Thus, no matter how many processors there are, the speedup is always less than 34.5.

Table 2 illustrates the speedups and efficiencies achieved for the matched flutter point evaluation of the SR3C-X2 propfan rotor using evaluation order (c) of Figure 4 and 7 processors executing concurrently. Fairly high speedups were achieved due to the large values of α and the modification of influence coefficient

evaluation order for better load balancing. As expected, the efficiency of this concurrent adaptation is higher when a larger number of panel rows were used in the analysis. However, impressive speedup and efficiency are obtained even when 9 panel rows were used.

Figure 5 illustrates the performance of this implementation using different numbers of processors. The ideal and theoretical speedups are also shown for comparison. The theoretical speedup line is close to the ideal speedup line because the bulk of the computational effort is spent in the parallelized portion of the analysis. For both the 9 panel rows and the 17 panel rows, the actual speedup line deviates more from the theoretical and ideal speedup lines for larger number of processors primarily because of longer processor idling.

Clearly, speedup and efficiency would be improved when the number of computations for the aerodynamic coefficients increases in comparison to the rest of the analysis. This would be the case, for example, if the aeroelastic analysis is performed in the supersonic regime, or if finer paneling is required for analysis. Figure 5 illustrates that the theoretical speedups as well as the actual speedups were higher for the case of 17 panel rows (Figure 5 (b)) than for that for 9 panel rows (Figure 5 (a)). Table 2 shows that, when the number of panel rows was increased from 9 to 17, the speedup improved 6 percent from 4.98 to 5.28 for 7 processors.

Conversely, speedup and efficiency of the parallel implementation would decrease if the number of computations for the aerodynamic coefficients decreases in comparison to the rest of the analysis. This is the case when the analysis uses a larger number of modes. For example, Table 2 shows that, when the number of modes is increased from 2 to 6, the fraction of the program executed in parallel decreased from 0.971 to 0.910 for 9 panel rows. This resulted in the speedup decreasing 15 percent from 4.98 to 4.24 for 7 processors.

Gains in speedup and efficiency, similar to those obtained in the current implementation, can also be expected in other aeroelastic analysis procedures, using the same parallel computational strategy as used here. Adaptation of this strategy is straightforward for panel method formulations, e. g., Watkins et al (1959) and Morino (1980). This is because integral equations similar to eq. (2), relating unsteady pressure disturbances and normal velocities over the blade, are obtained in all panel methods, even though the kernel function itself may be substantially different.

Concluding Remarks

ASTROP3, an aeroelastic analysis program for propfans, was adapted and implemented in a shared memory concurrent processing environment and achieved efficiencies up to 75 percent using 7 processors. Only moderate modification of the corresponding sequential code was performed by using a high-level approach where parallel paths were identified in the computationally intensive portion of the sequential code and parallelized. The calculation of the unsteady aerodynamic coefficients was concurrentized and the independent concurrent subtasks were scheduled to reduce processor idle time and improve speedup and efficiency. The results obtained demonstrate the potential for parallelization of aeroelastic analysis procedures, particularly those using panel methods for calculating unsteady aerodynamic forces. The speedup and efficiency gained in the aerodynamic coefficient computation would also contribute to the overall speedup and efficiency of an automated multi-disciplinary design procedure of which the aeroelastic analysis would form a part.

References

Desmarias, R. N., 1982, "An Accurate and Efficient Method for Evaluating the Kernel of the Integral Equation Relating Pressure to Normal Wash in Unsteady Potential Flow", 23rd Structrues, Structural Dynamics and Materials Conference, New Orleans, LA, Vol. 2, pp. 243-255.

Kaza, K. R. V., Mehmed, O., Narayanan, G. V. and Murthy, D. V., 1989, "Analytical Flutter Investigation of a Composite Propfan Model", Journal of Aircraft, Vol. 26, No. 8, pp. 772-780.

Kaza, K. R. V., Williams, M. H., Mehmed, O. and Narayanan, G. V., 1988, "Aeroelastic Response of Metallic and Composite Propfan Models in Yawed Flow", NASA TM-100964.

Morino, L., 1980, "Steady, Oscillatory, and Unsteady Subsonic and Supersonic Aerodynamics -- Production Version (Soussa P1.1) Vol. 1 -- Theoretical Manual", NASA CR-159130.

Murthy, D. V. and Kaza, K. R. V., 1989, "A Computational Procedure for Automated Flutter Analysis", Communications in Applied Numerical Methods, Vol. 5, No. 1, pp. 29-37.

Noor, A. K. and Atluri, S. N., 1987, "Advances and Trends in Computational Structural Mechanics", AIAA Journal, Vol. 25, No. 7, pp. 977-995.

Ortega, J. M., 1988, Introduction to Parallel and Vector Solution of Linear Systems, Plenum Press, New York.

Watkins, et al 1959, "A Systematic Kernel Function Procedure for Determining Aerodynamic Forces on Oscillating or Steady Finite Wings at Subsonic Speeds", NACA TR R-48.

Williams, M. H. and Hwang, C., 1986, "Three Dimensional Unsteady Aerodynamics and Aeroelastic Response of Advanced Turboprops", presented at the AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference, San Antonio, TX.

Table 1. Average idle times using 7 processors.

Evaluation order	Idle time per processor (percent of total time)	
	9 radial panel rows	17 radial panel rows
(a)	9.6	2.9
(b)	1.1	0.6
(c)	2.3	0.6

Table 2. Speedup and Efficiency using 7 processors.

No. of radial panel rows	No. of modes	α	Theoretical speedup	Speedup achieved	Efficiency (percent)
9	2	0.971	5.96	4.98	71.1
9	6	0.910	4.55	4.24	60.6
17	2	0.977	6.15	5.28	75.4
17	6	0.939	5.12	4.81	68.7

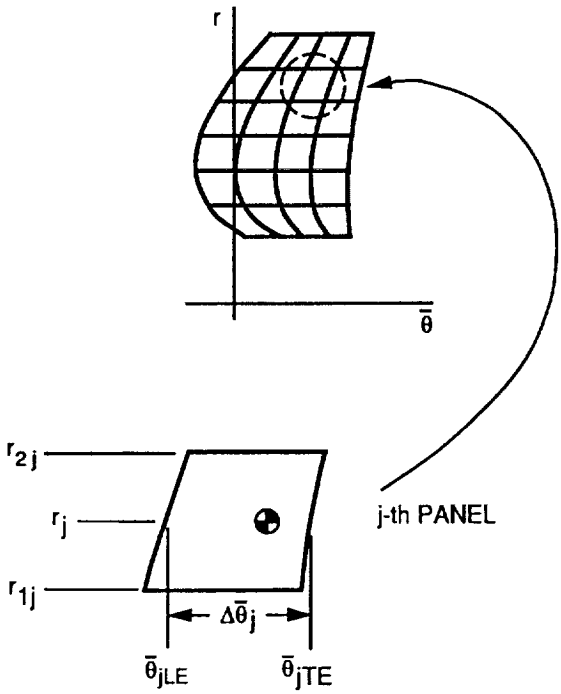

Figure 1. - Blade paneling.

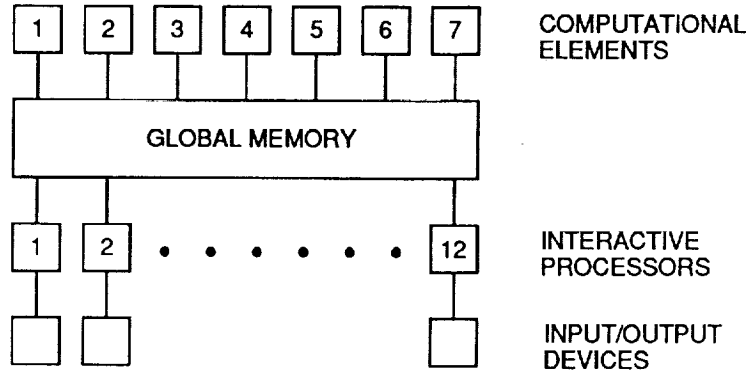

Figure 2. - System architecture.

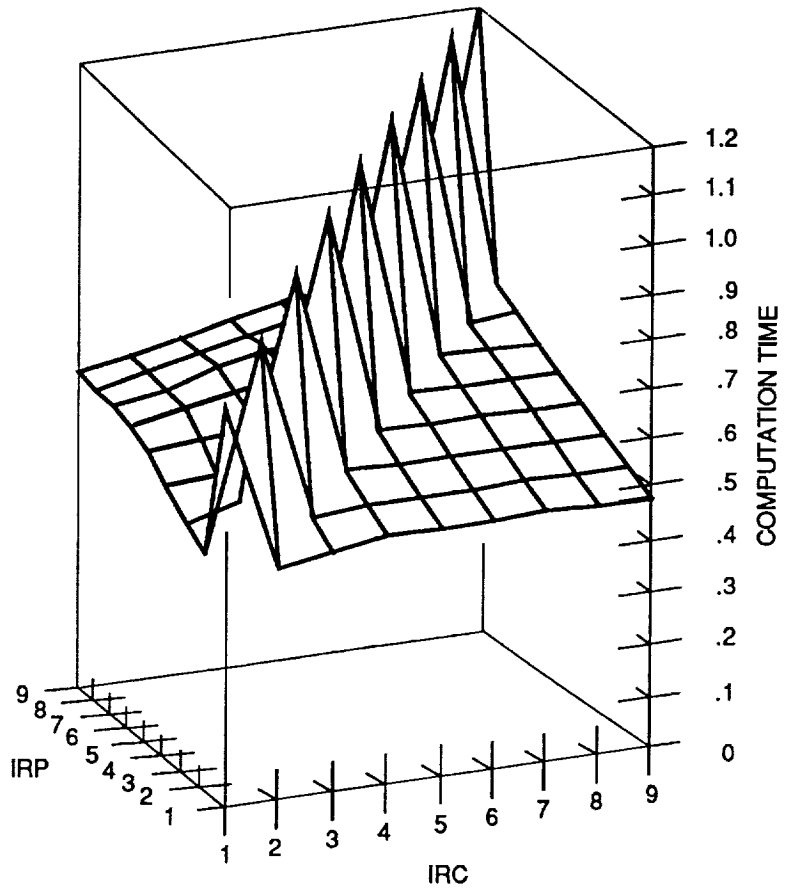

Figure 3. - Variation of calculation time for influence coefficients with control (IRC) and pressure (IRP) panel row numbers.

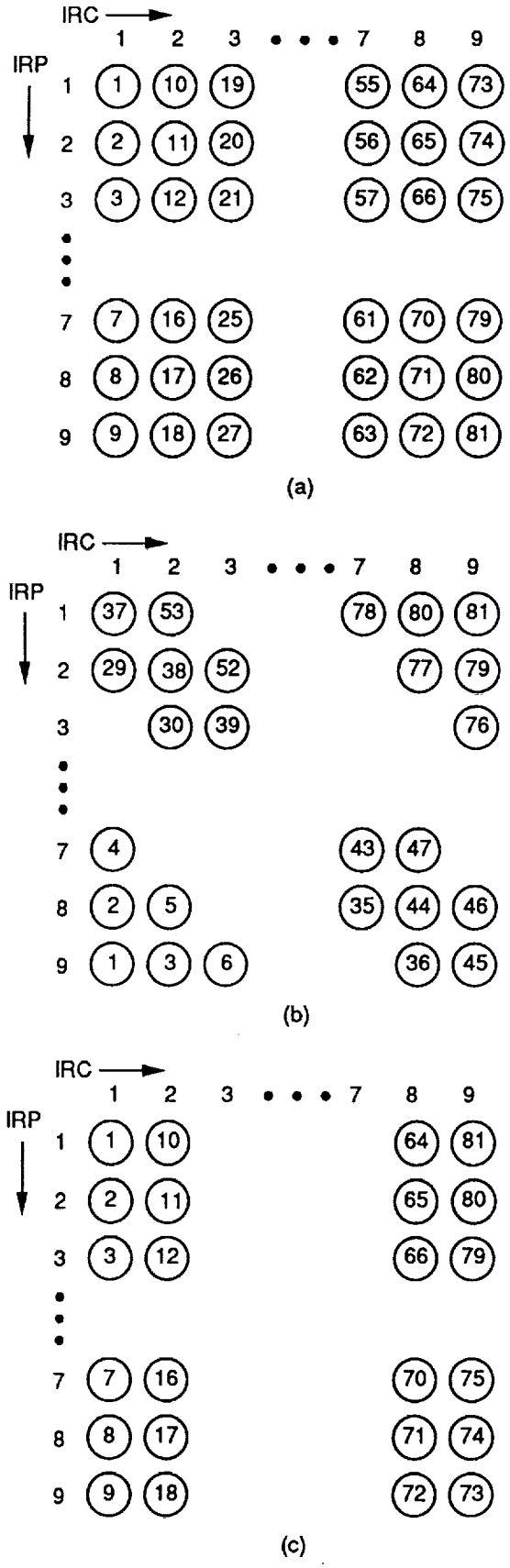

Figure 4. - Three evaluation orders of control (IRC) and pressure (IRP) panel row numbers in the calculation of influence coefficients.

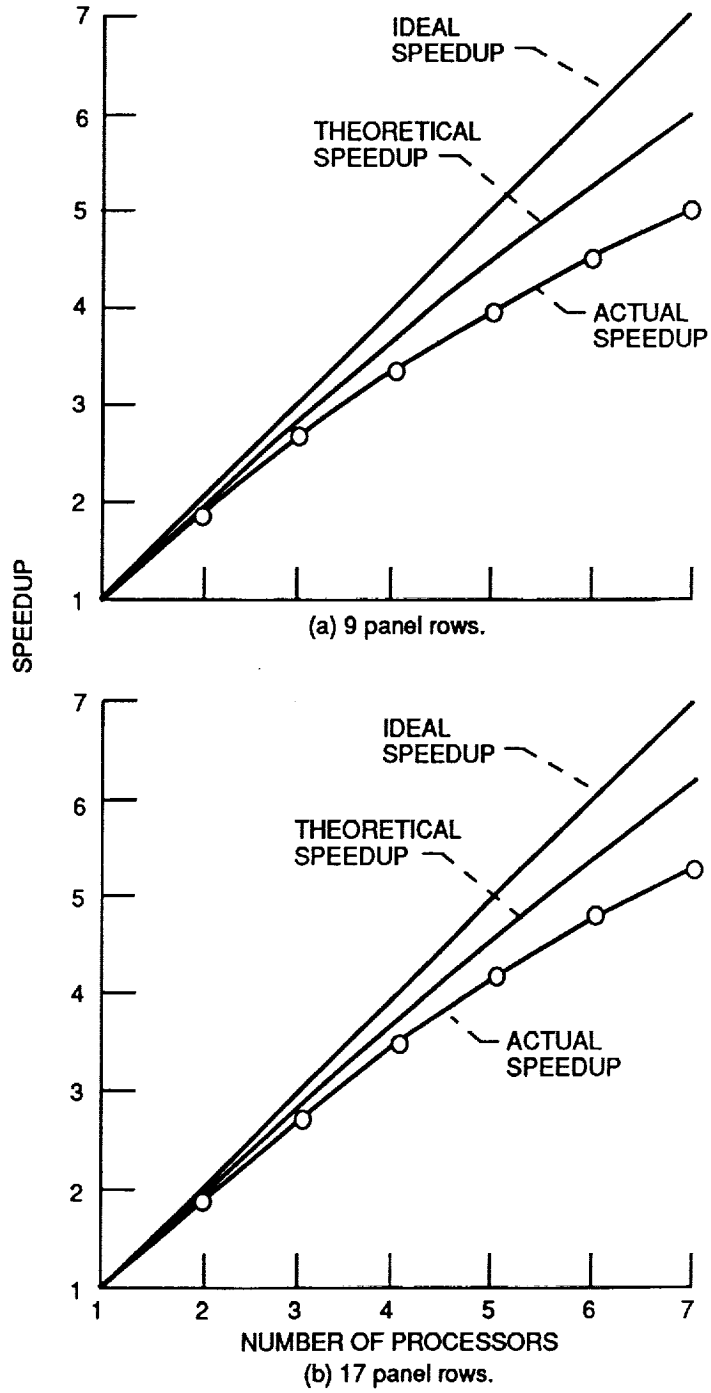

Figure 5. - Variation of actual and theoretical speedups with number of available processors.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102455		

4. Title and Subtitle	5. Report Date
Concurrent Processing Adaptation of Aeroelastic Analysis of Propfans	January 1990
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
David C. Janetzke and Durbha V. Murthy	E-5105
	10. Work Unit No.
	505-63-1B

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	
	13. Type of Report and Period Covered
	Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546-0001	

15. Supplementary Notes

David C. Janetzke, NASA Lewis Research Center; Durbha V. Murthy, University of Toledo, Toledo, Ohio 43606 (work performed under NASA Grant NAG3-742) and NASA Resident Research Associate at Lewis Research Center. Portions of this material will be presented at the 31st Structures, Structural Dynamics and Materials Conference cosponsored by the AIAA, ASME, ASCE, AHS, and ASC, Long Beach, California, April 2-4, 1990.

16. Abstract

This paper reports on a study involving the adaptation of an advanced aeroelastic analysis program to run concurrently on a shared memory multiple processor computer. The program uses a three-dimensional compressible unsteady aerodynamic model and blade normal modes to calculate aeroelastic stability and response of propfan blades. The identification of the computational parallelism within the sequential code and the scheduling of the concurrent subtasks to minimize processor idle time are discussed. Processor idle time in the calculation of the unsteady aerodynamic coefficients was reduced by the simple strategy of appropriately ordering the computations. Speedup and efficiency results are presented for the calculation of the matched flutter point of an experimental propfan model. The results show that efficiencies above 70 percent can be obtained using the present implementation with 7 processors. The parallel computational strategy described here is also applicable to other aeroelastic analysis procedures based on panel methods.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Concurrent processing Parallel processing Aeroelastic analysis Propfans	Unclassified - Unlimited Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	22	A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161